Fig. 1

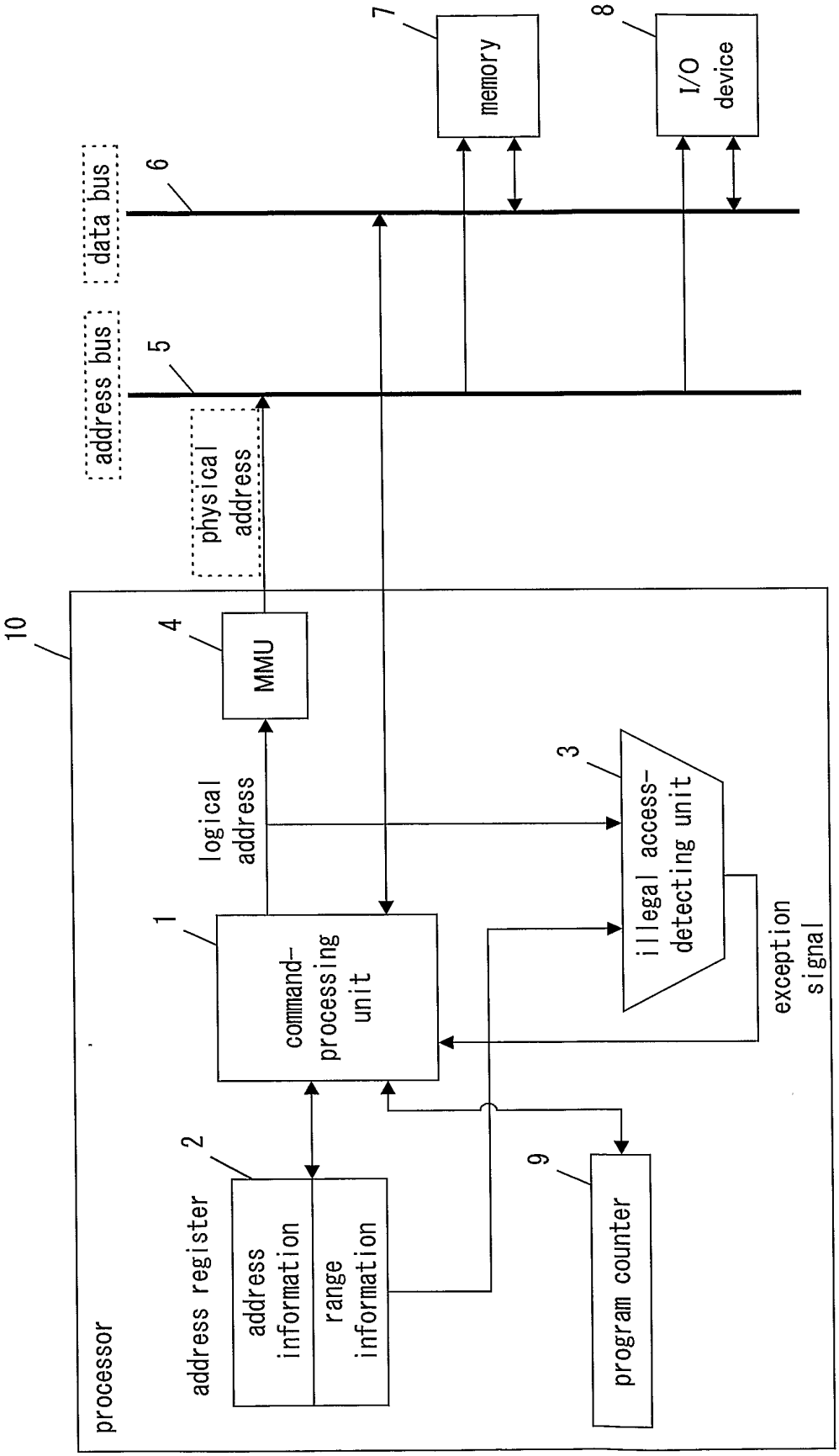
7
memory
8
I/O device
data bus
6
address bus
5
physical address
10
4
MMU
logical address
3
illegal access-detecting unit
1
command-processing unit
exception signal
2
address register
address information
range information
9
program counter
processor

Fig. 2(a)

```
01: int main()
02: {
03:     int a[3];
04:
05:     foo(a);
06:
07:     return 0;
08: }
09:
10: void foo(int *p)
11: {
12:     int i;
13:
14:     for ( i = 0; i < 4; i++ )
15:         *p++ = 0;
16: }
```

① compiler rcognizes size "3"

② generating pointer with range information from the size recognized in ①

③ receiving pointer with range information

④ illegal loop

Fig. 2(b)

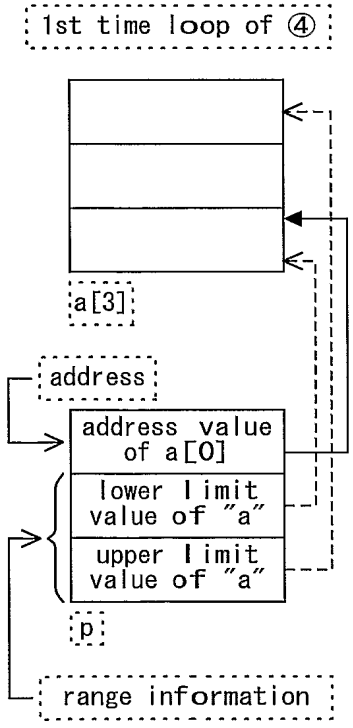

Fig. 2(c)

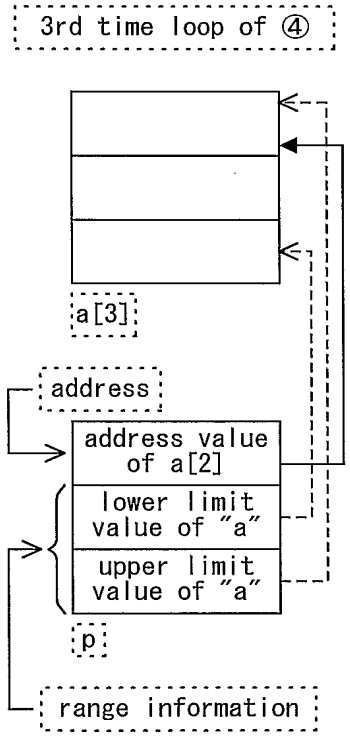

Fig. 2(d)

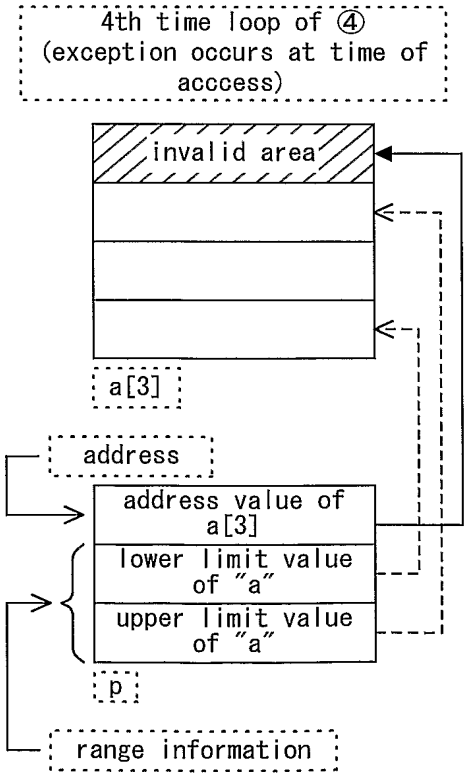

Fig. 3(a)

```
01: extern void input_str(char *str);
02:
03: int main()
04: {
05:     char buf[32];
06:     int  len = 0;
07:
08:     input_str(buf);
09:
10:     while ( buf[len] != 0)
11:         len++;
12:
13:     printf( "%dThis is a character.¥n" , len);
14:
15:     return 0;
16: }
```

source code (main.c)

Fig. 3(b)

```
01: void input_str(char *str)
02: {
03:       int c;
04:
05:       while((c = getchar()) != EOF)
06:           *str++ = c;
07: }
```

source code (input_int.c)

Fig. 3(c)

```
main.c:
      char[] UPPER          ;upper expanding
      *      FIXED          ;fixed size

input_str.c:
      *      FIXED          ;fixed size
```

failure recovering information file

Fig. 4(a)

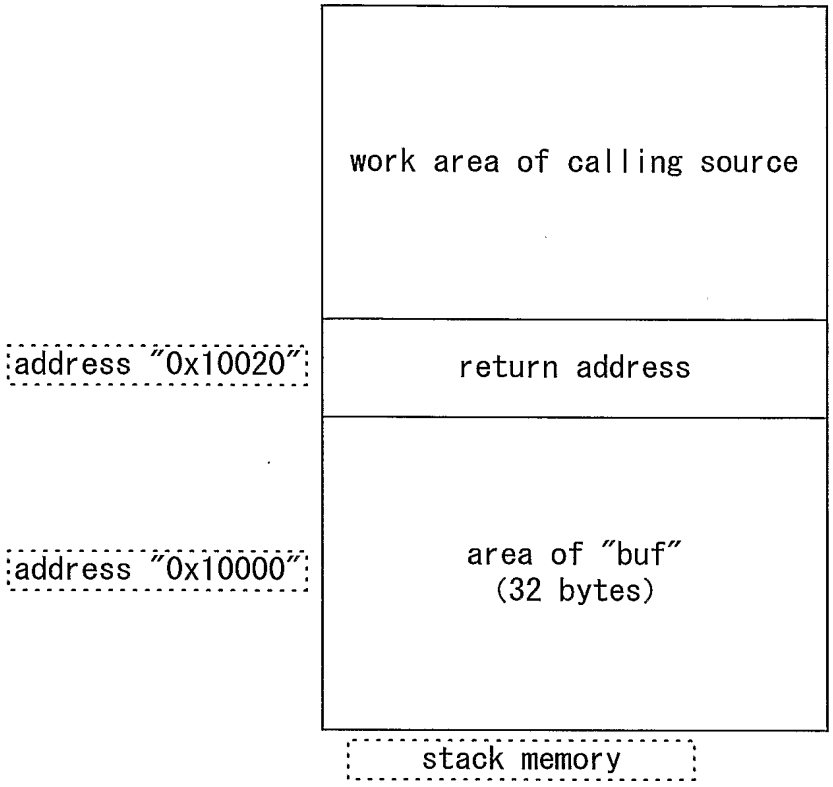
work area of calling source
address "0x10020"
return address
address "0x10000"
area of "buf"
(32 bytes)
stack memory

Fig. 4(b)

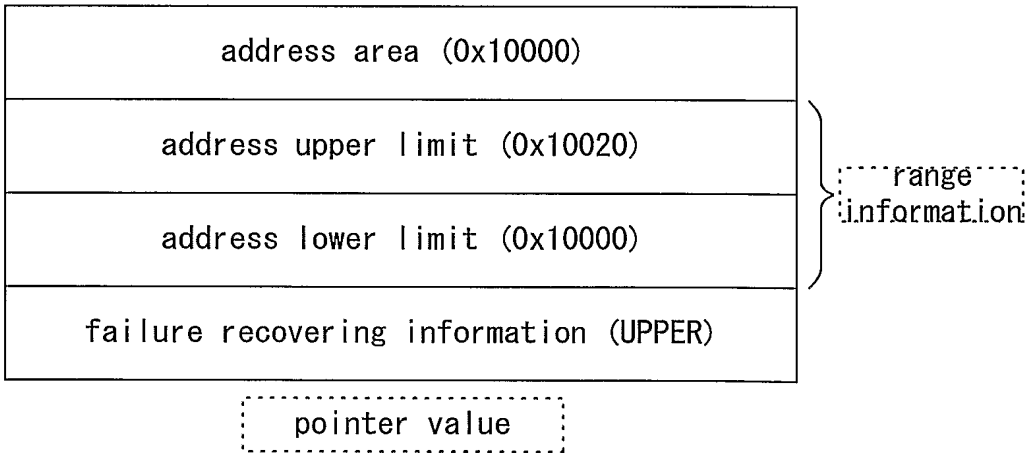
address area (0x10000)
address upper limit (0x10020)
address lower limit (0x10000)
failure recovering information (UPPER)
range information
pointer value

Fig. 5(a)

```
01: char a[5];
02:
03: extern void foo(char *p);
04:
05: int main()
06: {
07:       foo(&a[1]);
08:
09:       return 0;
10: }
```

source code in C language

Fig. 5(b)

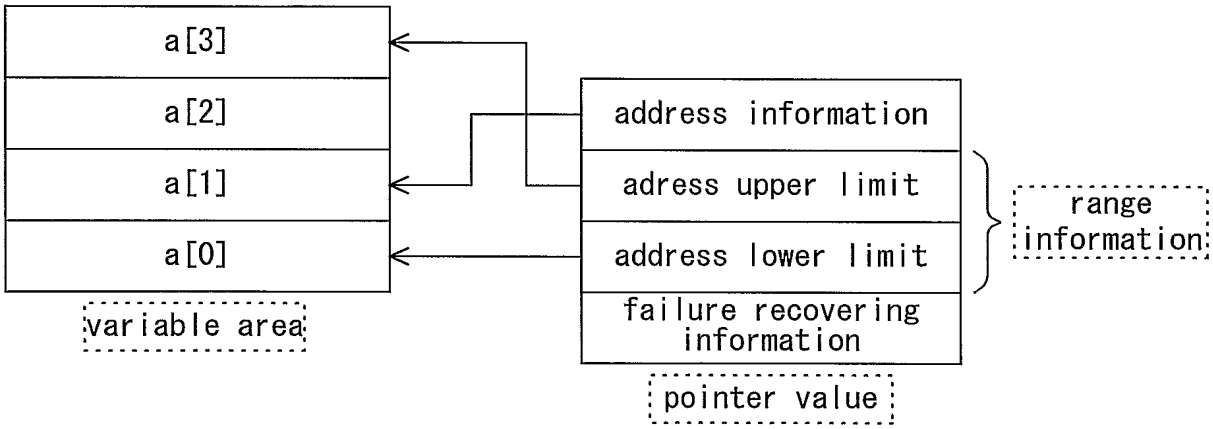

Fig. 6

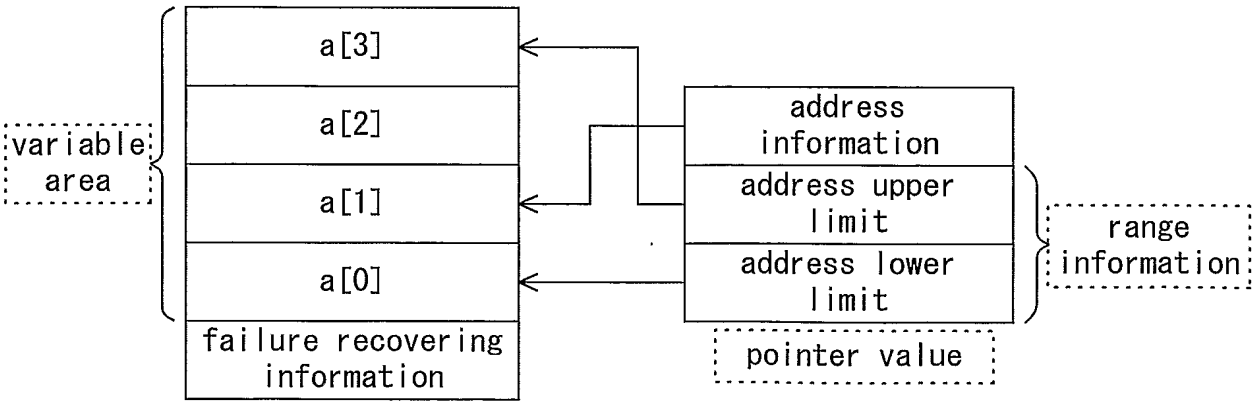

Fig. 7

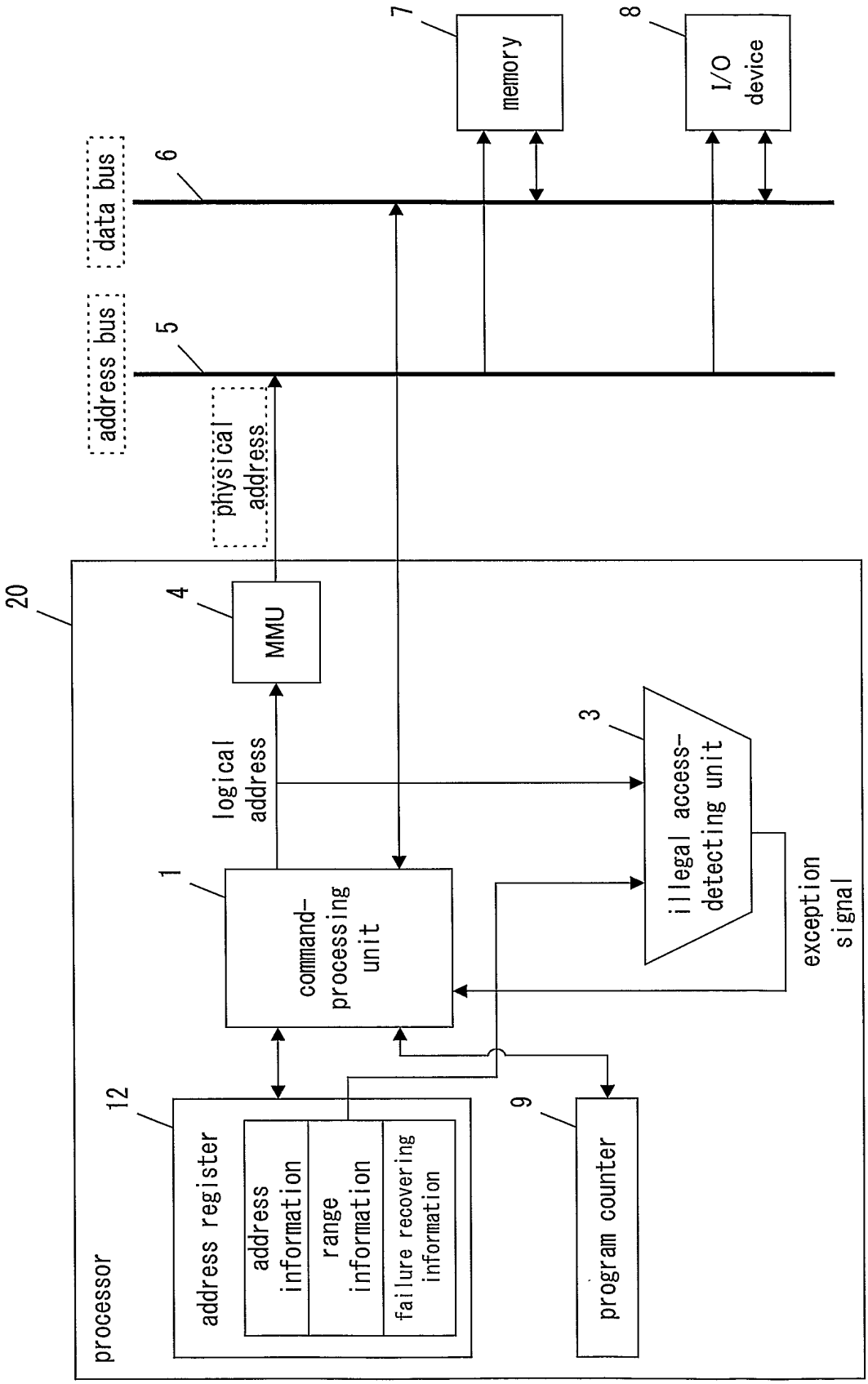
7
memory
8
I/O device
data bus
6
address bus
5
physical address
20
4
MMU
logical address
3
illegal access-detecting unit
1
command-processing unit
exception signal
12
address register
address information
range information
failure recovering information
9
program counter
processor

Fig. 8

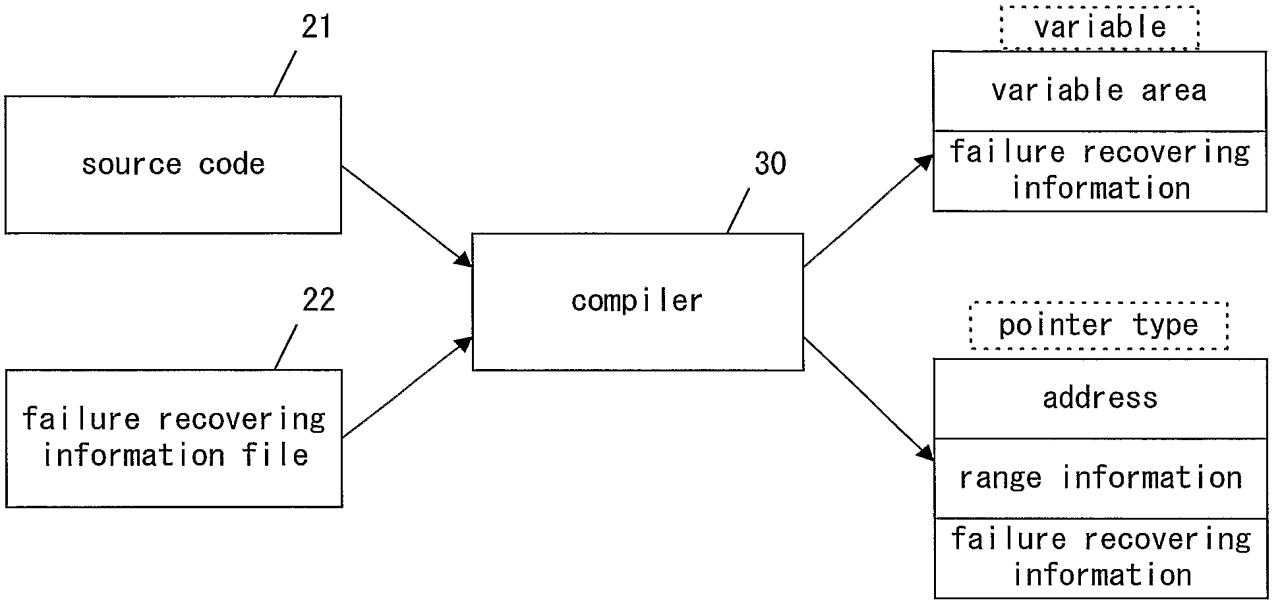
21
source code
22
failure recovering information file
compiler
30
variable
variable area
failure recovering information
pointer type
address
range information
failure recovering information

Fig. 9

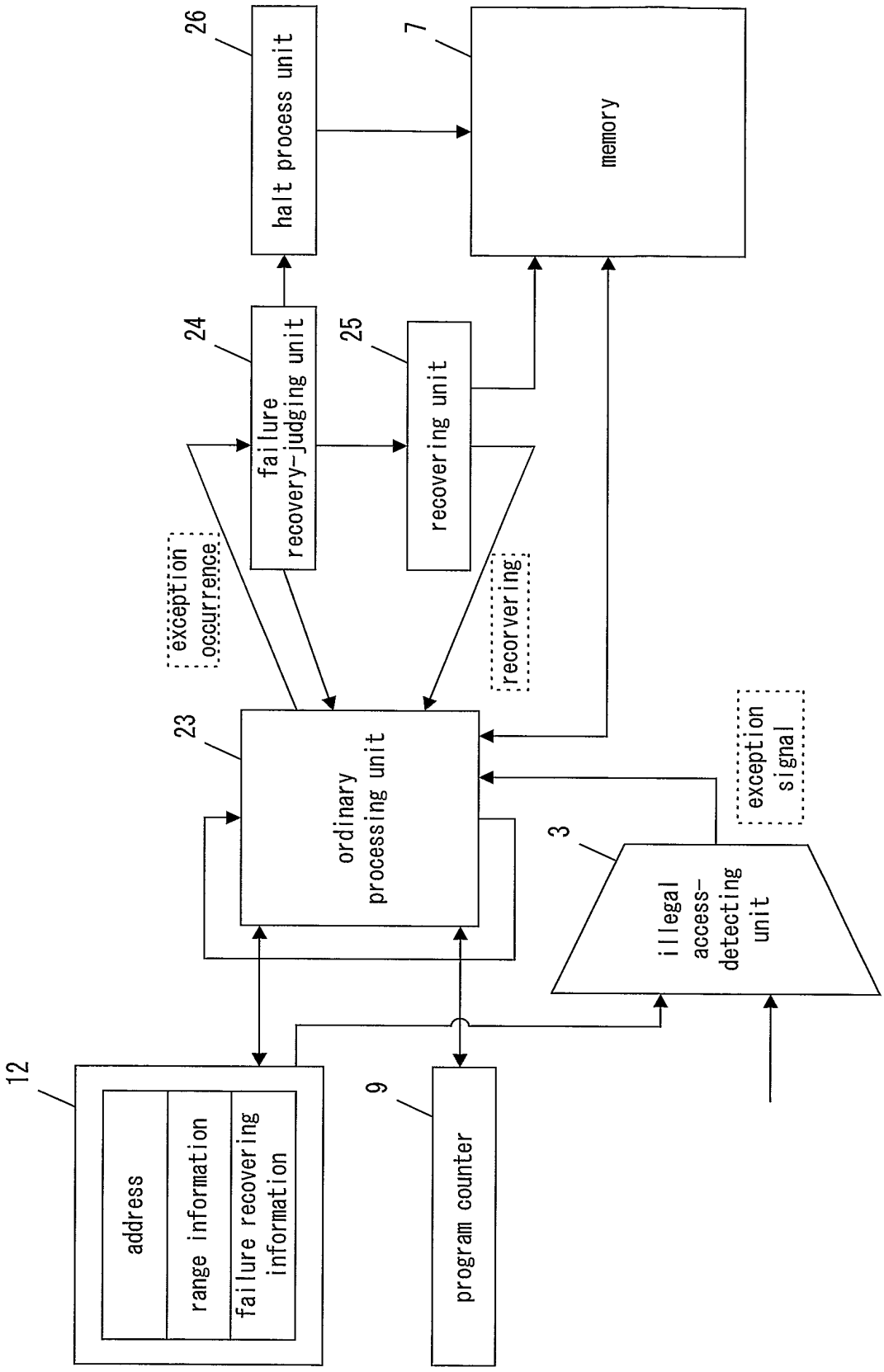
26
halt process unit
7
memory
24
failure recovery-judging unit
25
recovering unit
exception occurrence
recorverring
23
ordinary processing unit
exception signal
3
illegal access-detecting unit
12
address
range information
failure recovering information
9
program counter

Fig. 10

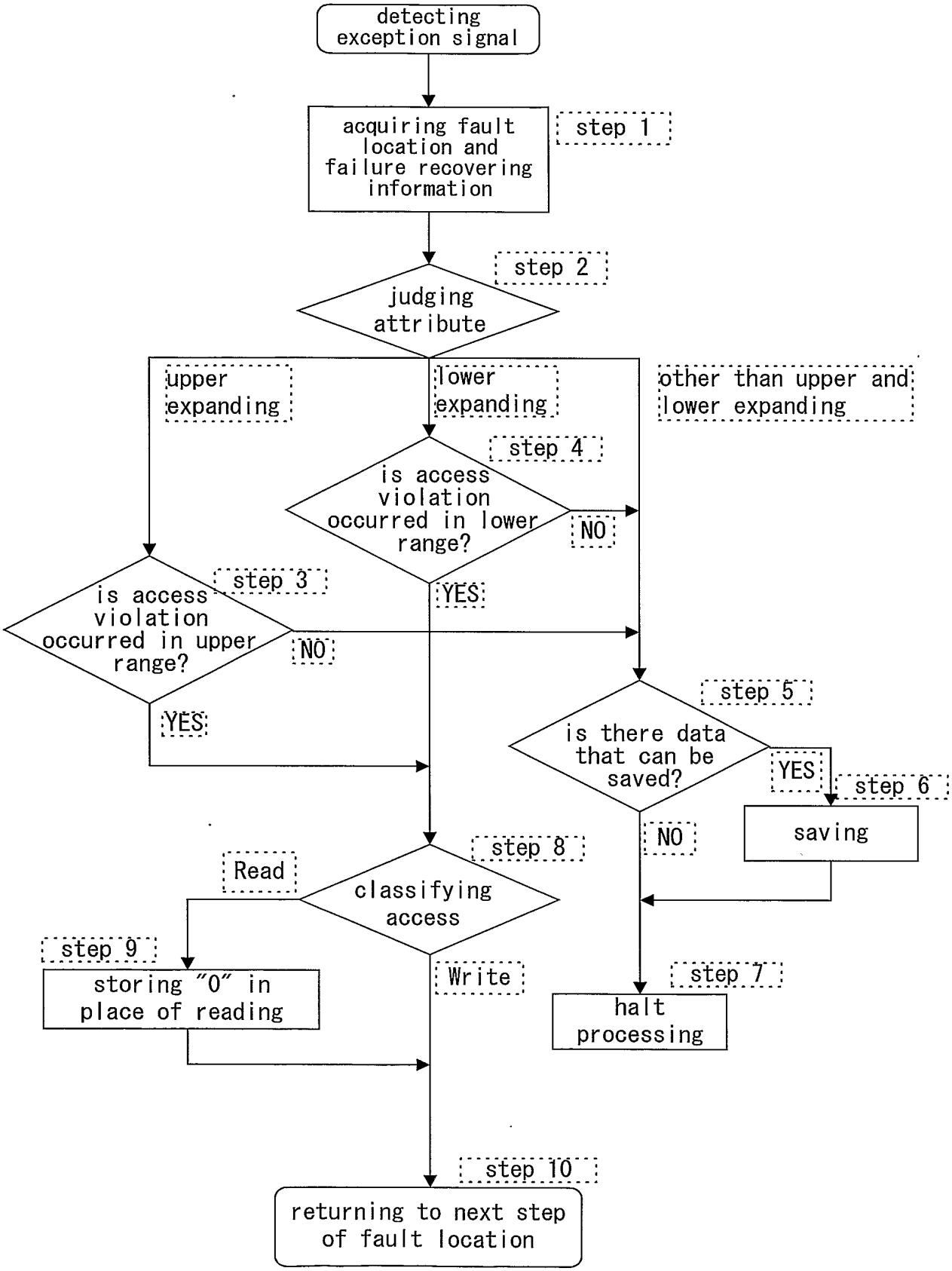
detecting exception signal
acquiring fault location and failure recovering information
step 1
step 2
judging attribute
upper expanding
lower expanding
other than upper and lower expanding
step 4
is access violation occurred in lower range?
NO
YES
step 3
is access violation occurred in upper range?
NO
YES
step 5
is there data that can be saved?
YES
step 6
saving
NO
step 8
Read
classifying access
step 9
storing "0" in place of reading
Write
step 7
halt processing
step 10
returning to next step of fault location

Fig. 11

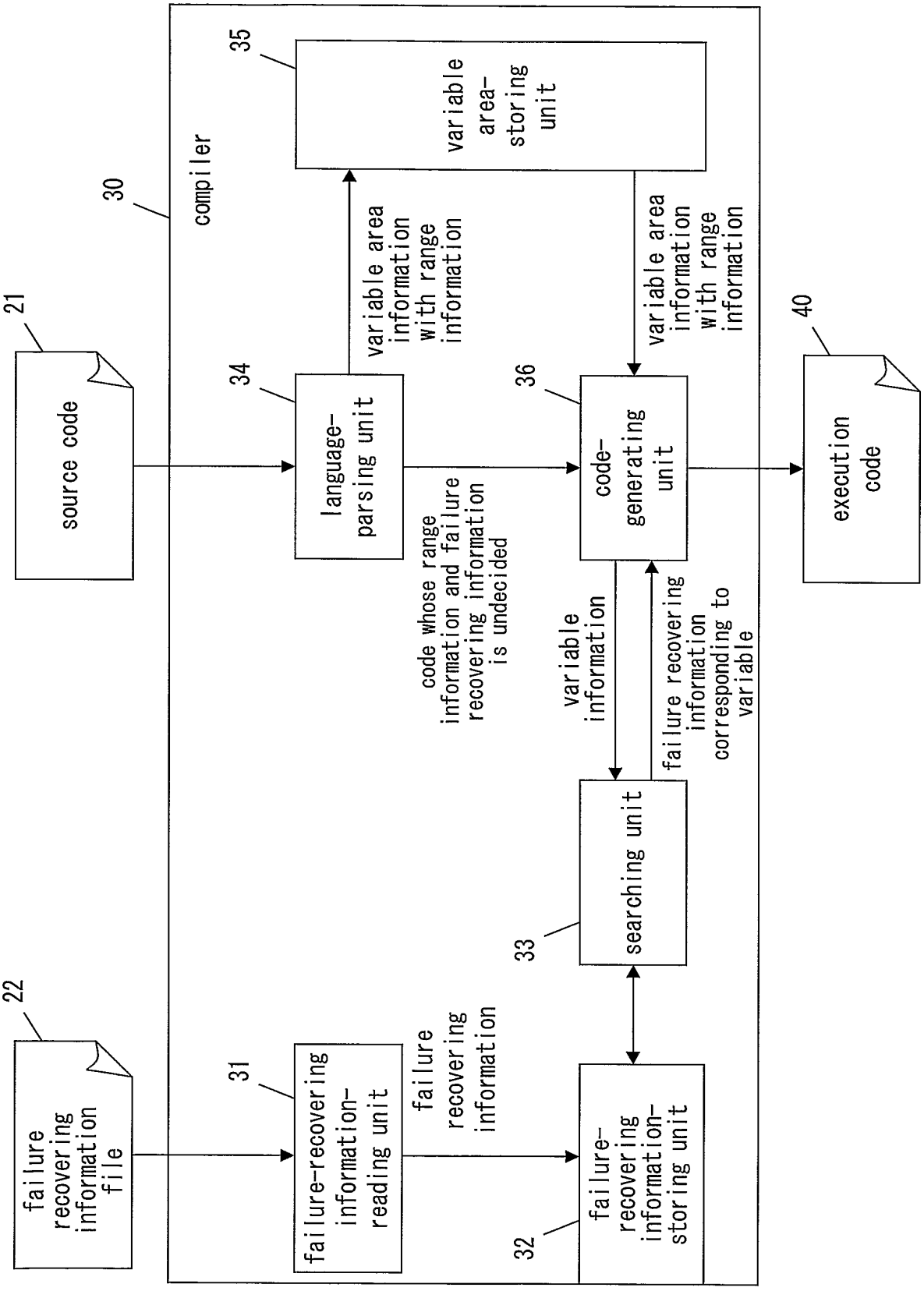
compiler
30
source code
21
failure recovering information file
22
language-parsing unit
34
failure-recovering information-reading unit
31
code-generating unit
36
searching unit
33
failure-recovering information-storing unit
32
variable area-storing unit
35
execution code
40
variable area information with range information
variable area information with range information
code whose range information and failure recovering information is undecided
failure recovering information
variable information
failure recovering information corresponding to variable

Fig. 12

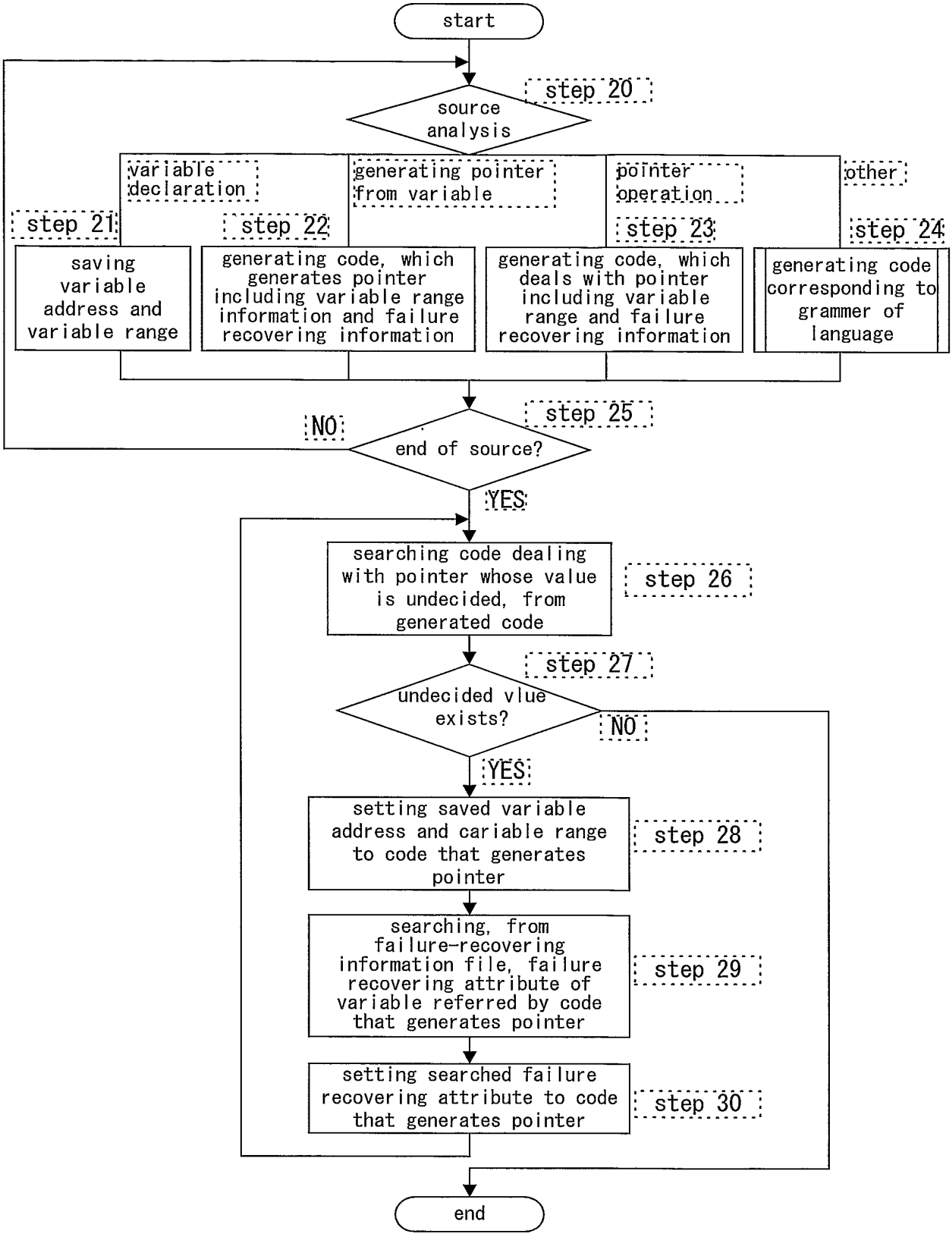
start
step 20
source analysis
variable declaration
generating pointer from variable
pointer operation
other
step 21
step 22
step 23
step 24
saving variable address and variable range
generating code, which generates pointer including variable range information and failure recovering information
generating code, which deals with pointer including variable range and failure recovering information
generating code corresponding to grammer of language
step 25
NO
end of source?
YES
searching code dealing with pointer whose value is undecided, from generated code
step 26
step 27
undecided vlue exists?
NO
YES
setting saved variable address and cariable range to code that generates pointer
step 28
searching, from failure-recovering information file, failure recovering attribute of variable referred by code that generates pointer
step 29
setting searched failure recovering attribute to code that generates pointer
step 30
end

Fig. 13

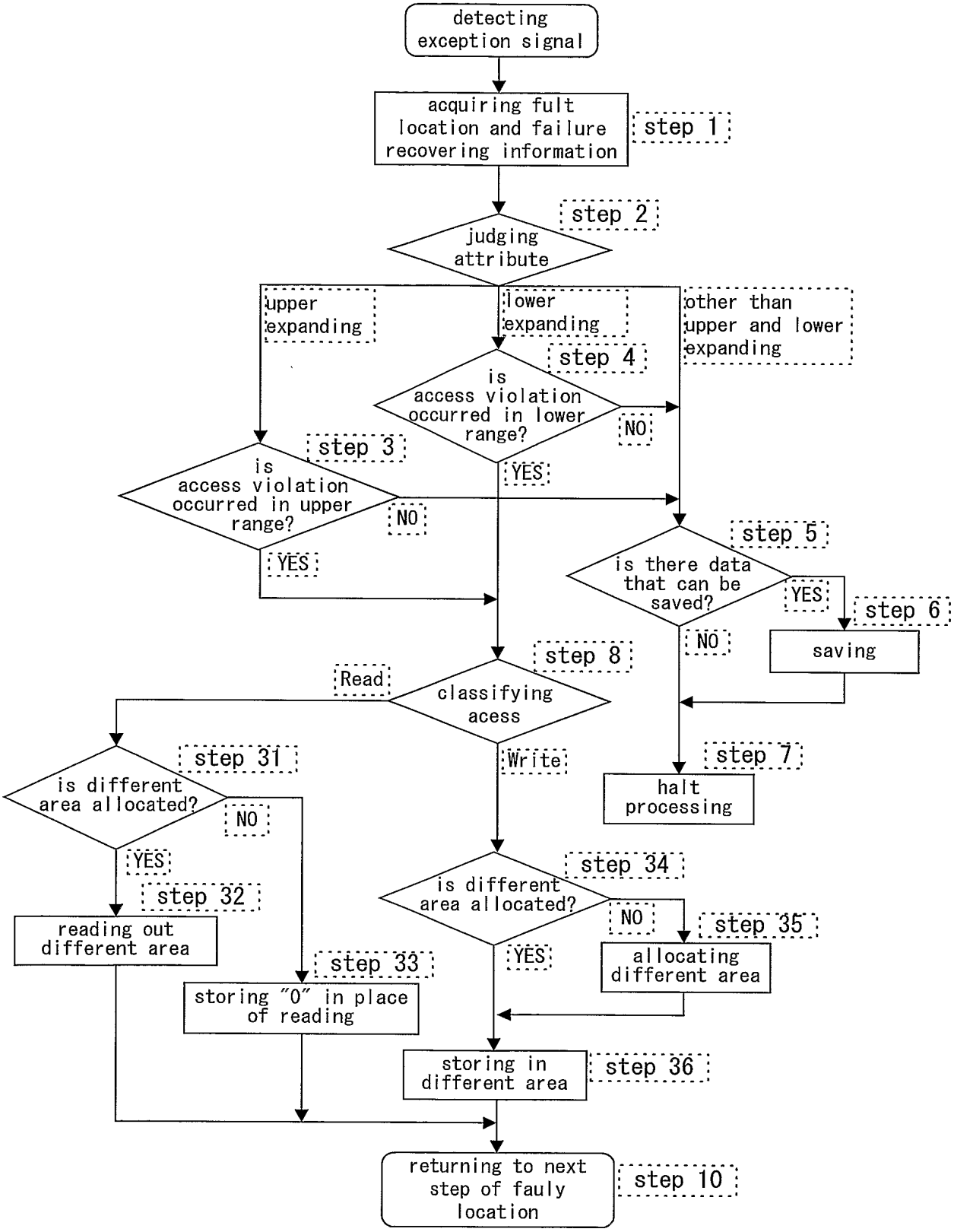
detecting exception signal
acquiring fult location and failure recovering information
step 1
step 2
judging attribute
upper expanding
lower expanding
other than upper and lower expanding
step 4
is access violation occurred in lower range?
NO
YES
step 3
is access violation occurred in upper range?
NO
YES
step 5
is there data that can be saved?
YES
step 6
saving
NO
step 8
Read
classifying acess
Write
step 7
halt processing
step 31
is different area allocated?
NO
YES
step 32
reading out different area
step 33
storing "0" in place of reading
step 34
is different area allocated?
NO
YES
step 35
allocating different area
storing in different area
step 36
returning to next step of fauly location
step 10

Fig. 14

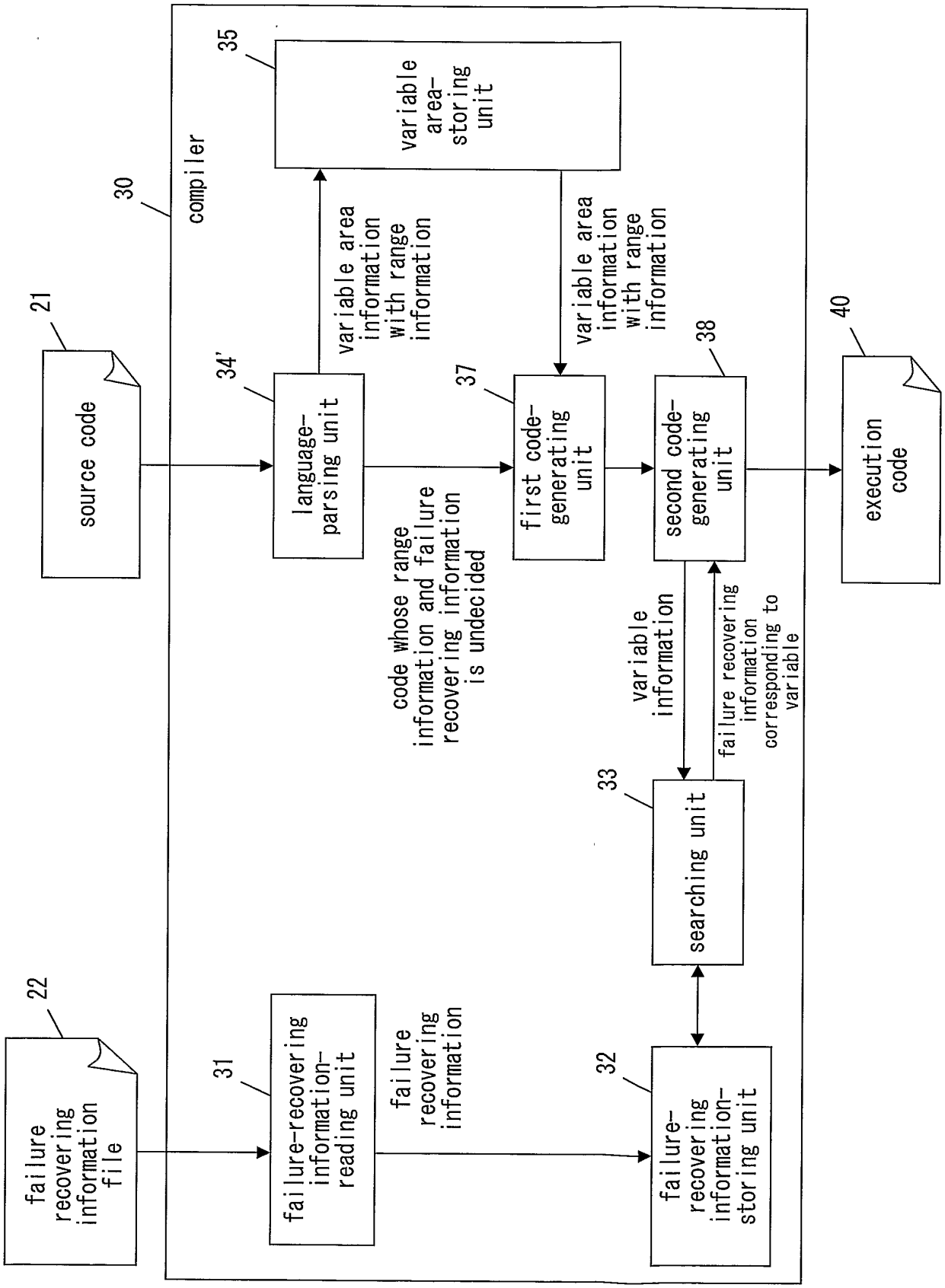
compiler
30
source code
21
language-parsing unit
34'
variable area information with range information
variable area-storing unit
35
variable area information with range information
first code-generating unit
37
second code-generating unit
38
execution code
40
code whose range information and failure recovering information is undecided
variable information
failure recovering information corresponding to variable
searching unit
33
failure recovering information
failure-recovering information-reading unit
31
failure-recovering information-storing unit
32
failure recovering information file
22

Fig. 15

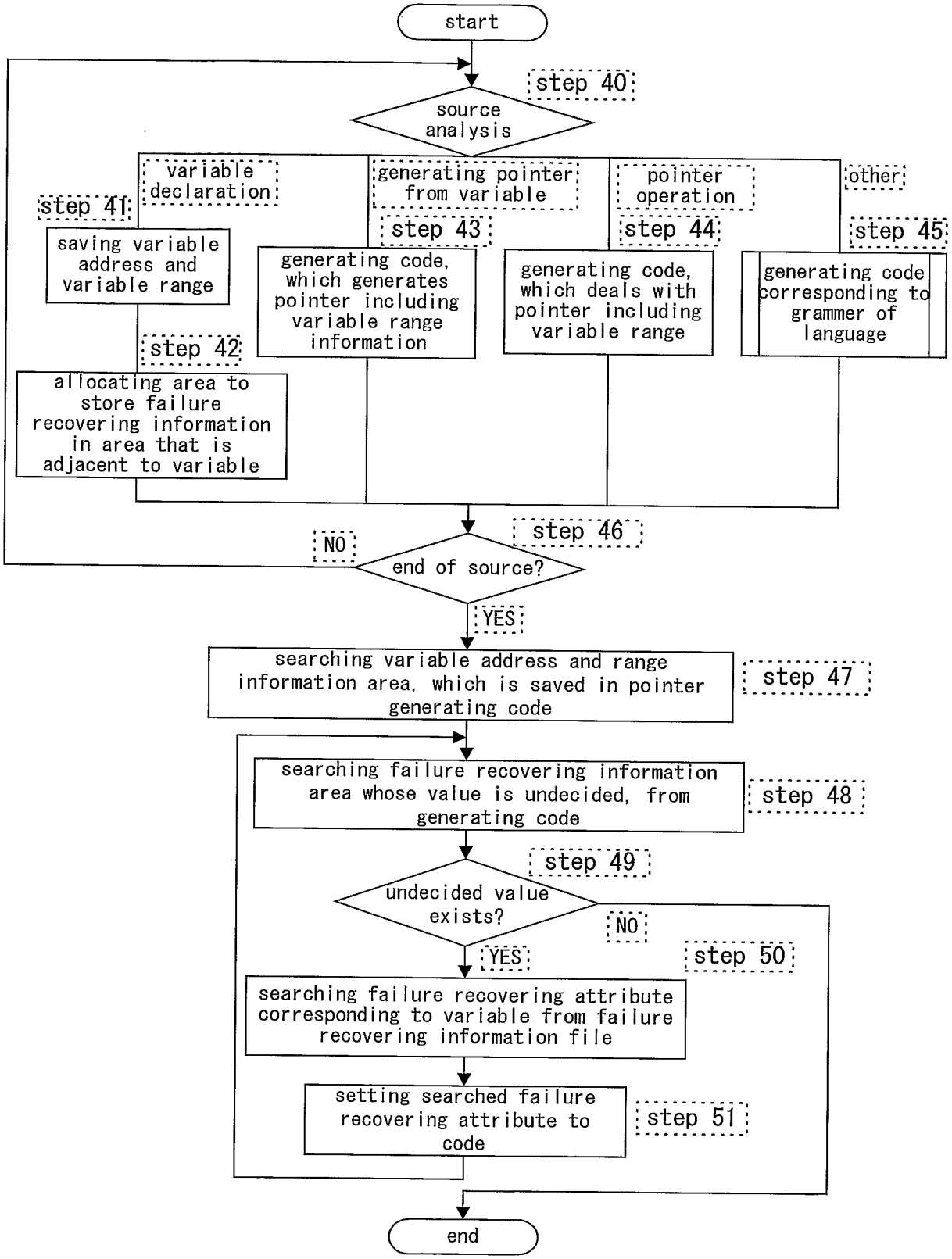
start
step 40
source analysis
variable declaration
generating pointer from variable
pointer operation
other
step 41
step 43
step 44
step 45
saving variable address and variable range
generating code, which generates pointer including variable range information
generating code, which deals with pointer including variable range
generating code corresponding to grammer of language
step 42
allocating area to store failure recovering information in area that is adjacent to variable
step 46
NO
end of source?
YES
searching variable address and range information area, which is saved in pointer generating code
step 47
searching failure recovering information area whose value is undecided, from generating code
step 48
step 49
undecided value exists?
NO
YES
step 50
searching failure recovering attribute corresponding to variable from failure recovering information file
setting searched failure recovering attribute to code
step 51
end

Fig. 16

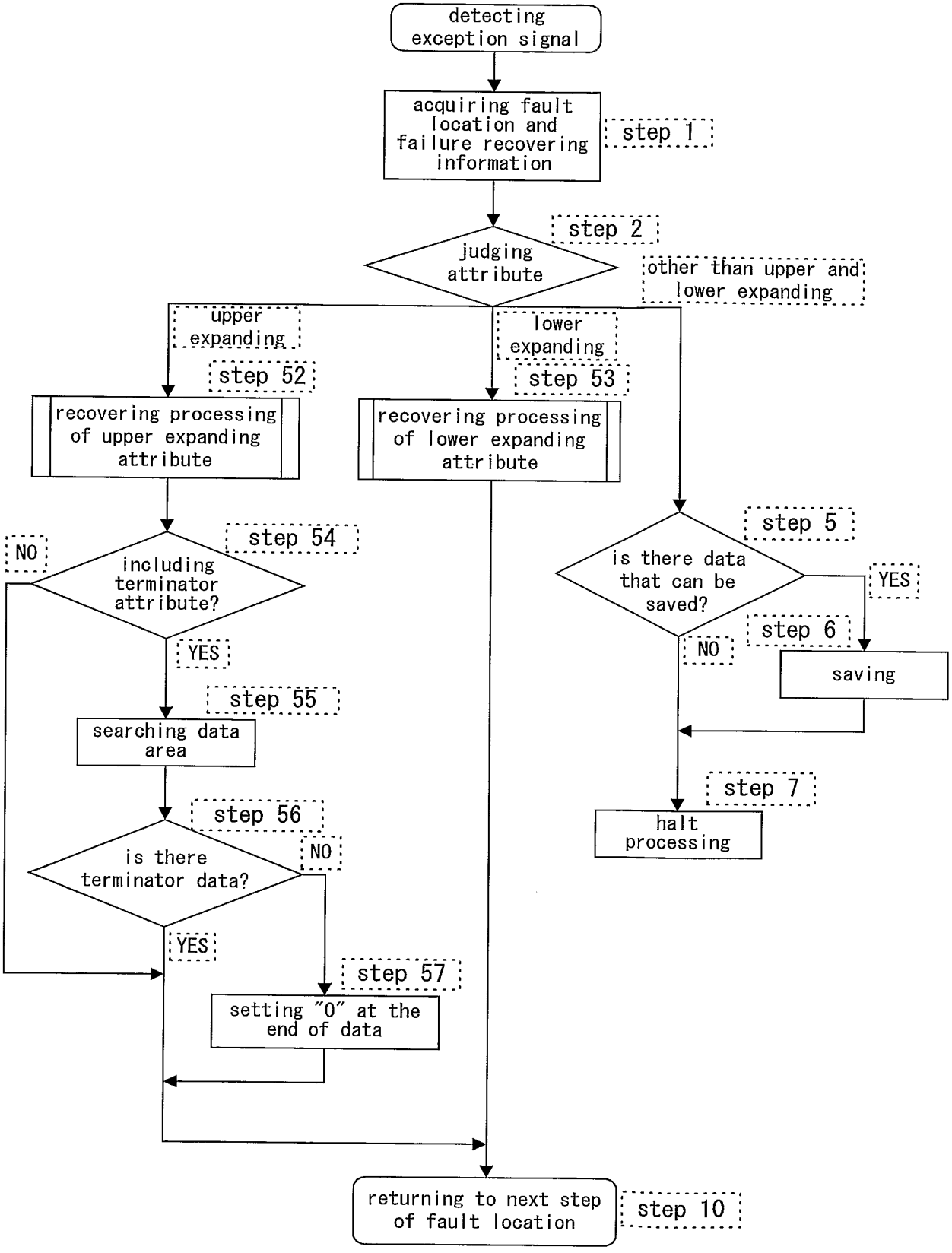
detecting exception signal
acquiring fault location and failure recovering information
step 1
judging attribute
step 2
other than upper and lower expanding
upper expanding
lower expanding
step 52
step 53
recovering processing of upper expanding attribute
recovering processing of lower expanding attribute
step 5
is there data that can be saved?
YES
NO
step 6
saving
step 7
halt processing
NO
including terminator attribute?
step 54
YES
step 55
searching data area
step 56
is there terminator data?
NO
YES
step 57
setting "0" at the end of data
returning to next step of fault location
step 10

Fig. 17

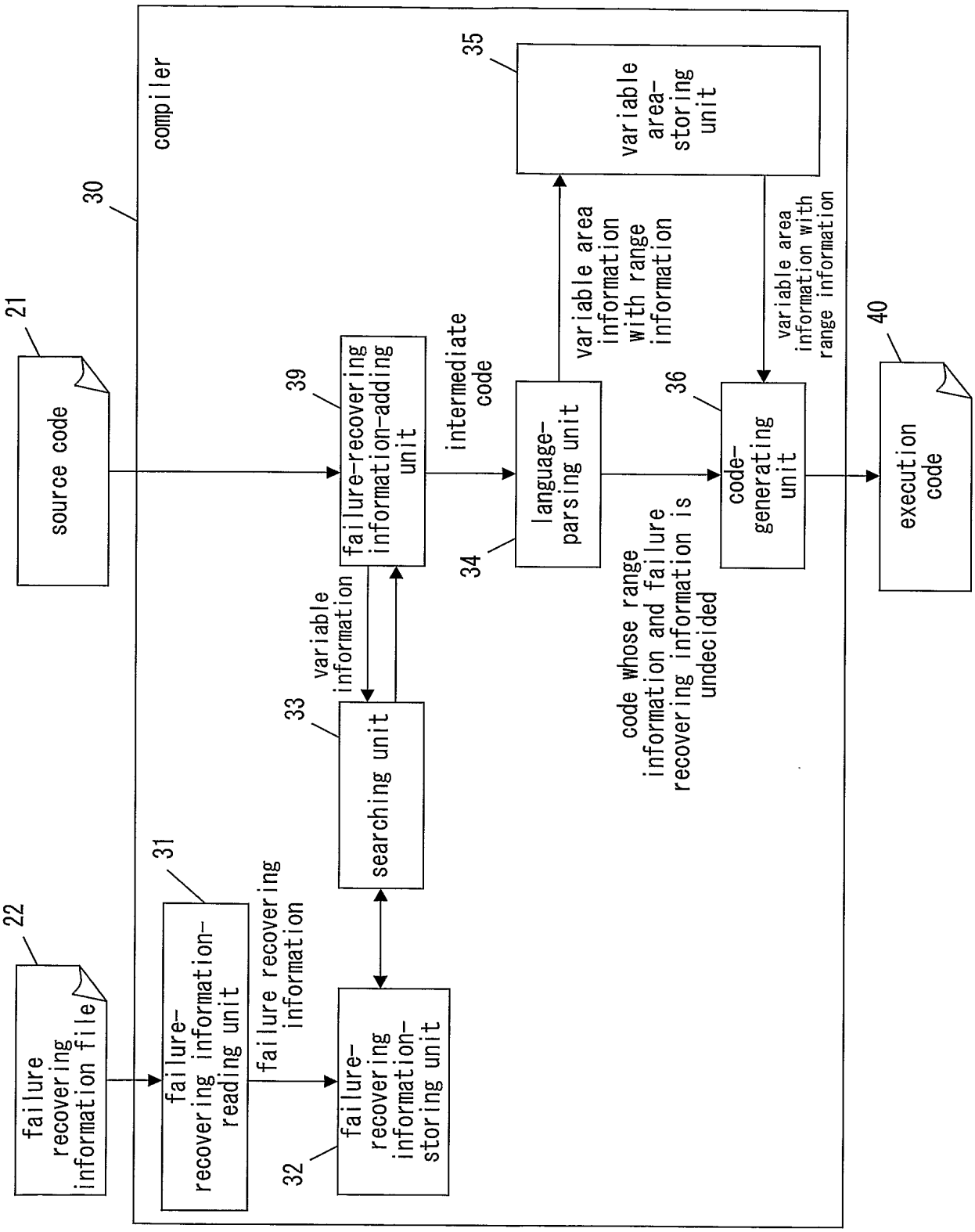
compiler
30
source code
21
failure recovering information file
22
failure-recovering information-reading unit
31
failure recovering information
failure-recovering information-storing unit
32
searching unit
33
variable information
failure-recovering information-adding unit
39
intermediate code
language-parsing unit
34
variable area information with range information
variable area-storing unit
35
code whose range information and failure recovering information is undecided
code-generating unit
36
variable area information with range information
execution code
40

Fig. 18

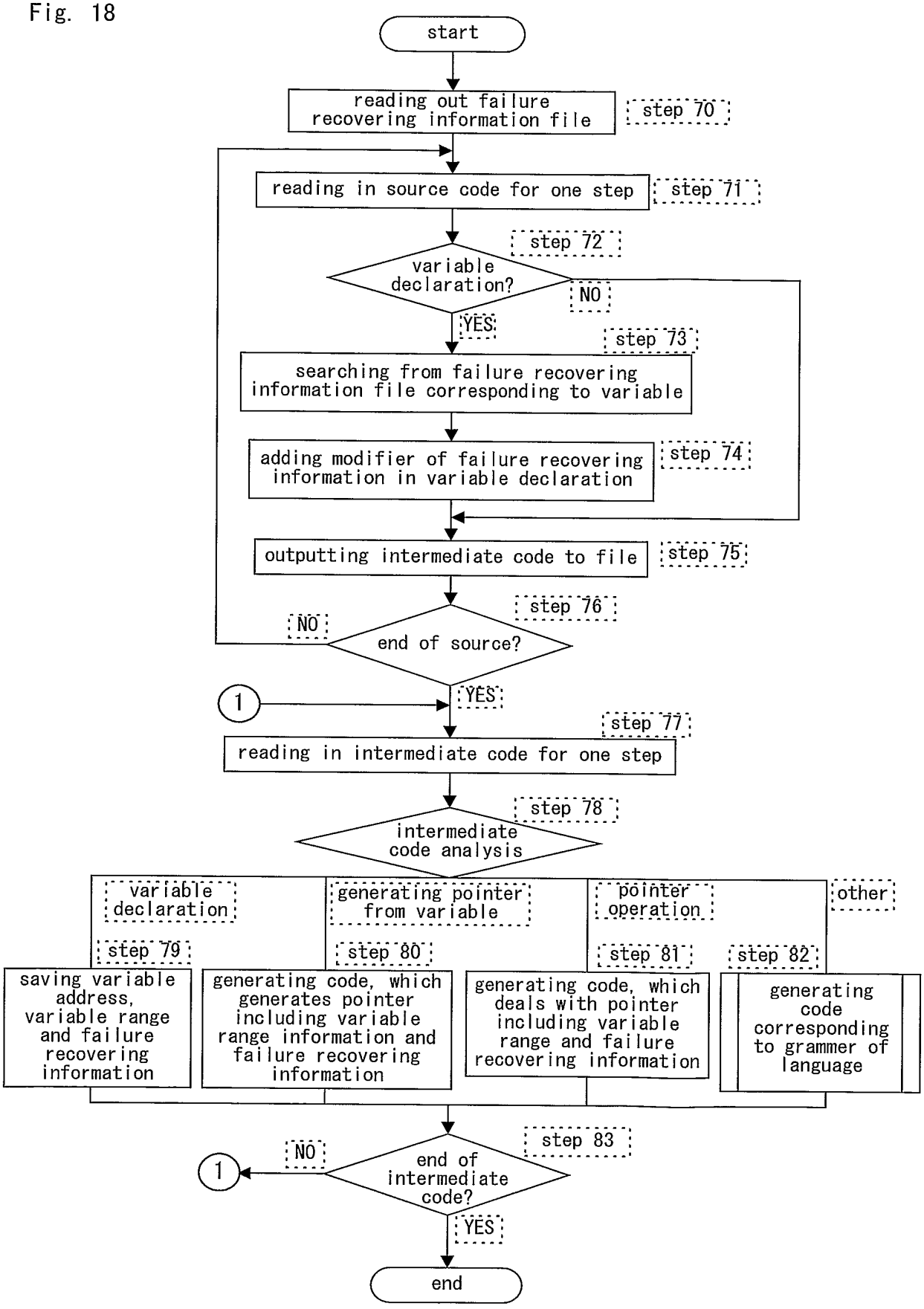
start
reading out failure recovering information file
step 70
reading in source code for one step
step 71
step 72
variable declaration?
NO
YES
step 73
searching from failure recovering information file corresponding to variable
adding modifier of failure recovering information in variable declaration
step 74
outputting intermediate code to file
step 75
step 76
NO
end of source?
1
YES
step 77
reading in intermediate code for one step
step 78
intermediate code analysis
variable declaration
generating pointer from variable
pointer operation
other
step 79
step 80
step 81
step 82
saving variable address, variable range and failure recovering information
generating code, which generates pointer including variable range information and failure recovering information
generating code, which deals with pointer including variable range and failure recovering information
generating code corresponding to grammer of language
step 83
NO
1
end of intermediate code?
YES
end

Fig. 19(a)

```
01: extern void input_str(char *str);
02:
03: int main()
04: {
05:     UPPER char buf[32];
06:     FIXED int  len=0;
07:
08:     input_str(buf);
09:
10:     while ( buf[len] ! = 0)
11:         len++;
12:
13:     printf( "%dThis is a character¥n" ,
len);
14:
15:     return 0;
16: }
```

intermediate code (after processing of "main.c")

Fig. 19(b)

```
01: void input_str(char *str);
02: {
03:       FIXED int c;
04:
05:       while( (c = getchar()) ! = EOF)
06:           *str++=c;
07: }
```

intermediate code (input_int.c)

Fig. 20

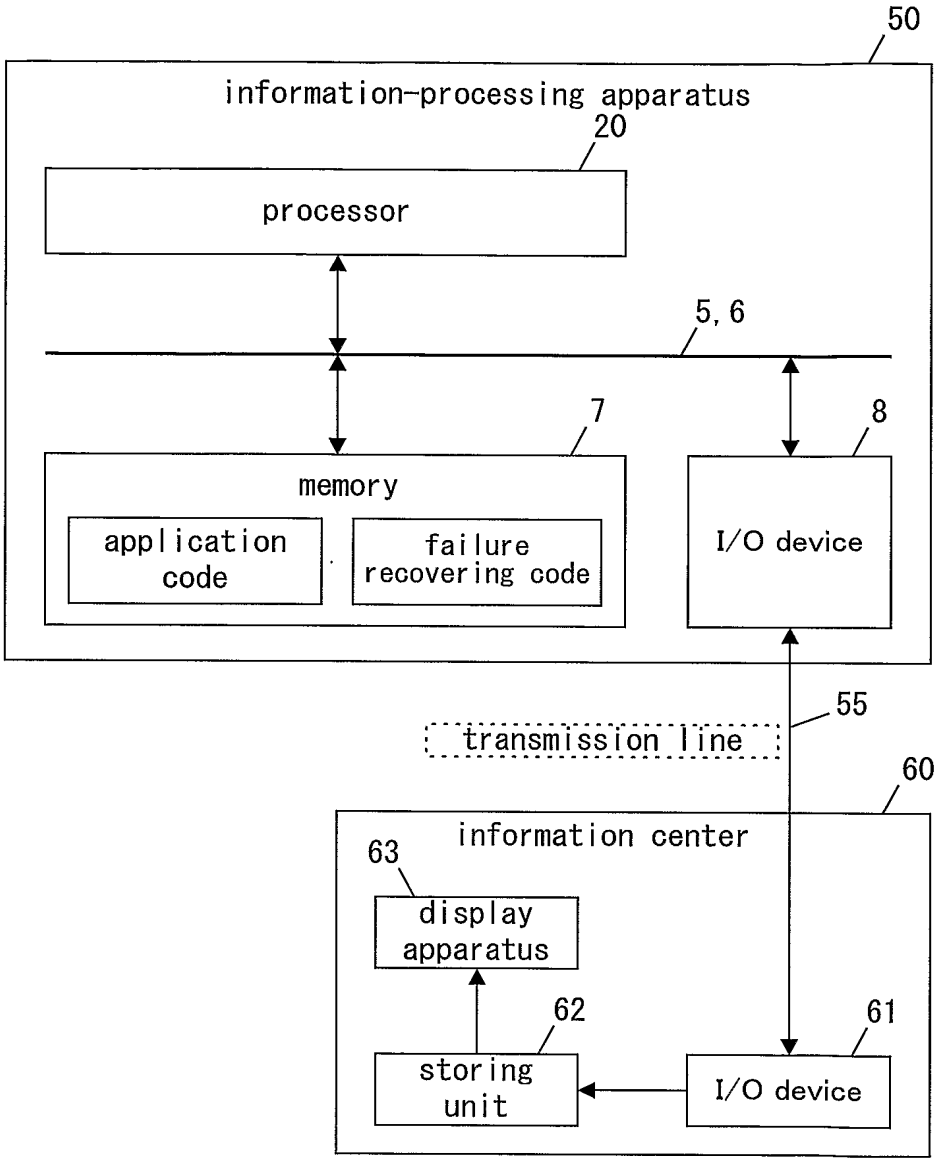
50
information-processing apparatus
20
processor
5, 6
7
8
memory
application code
failure recovering code
I/O device
55
transmission line
60
information center
63
display apparatus
62
61
storing unit
I/O device

Fig. 21

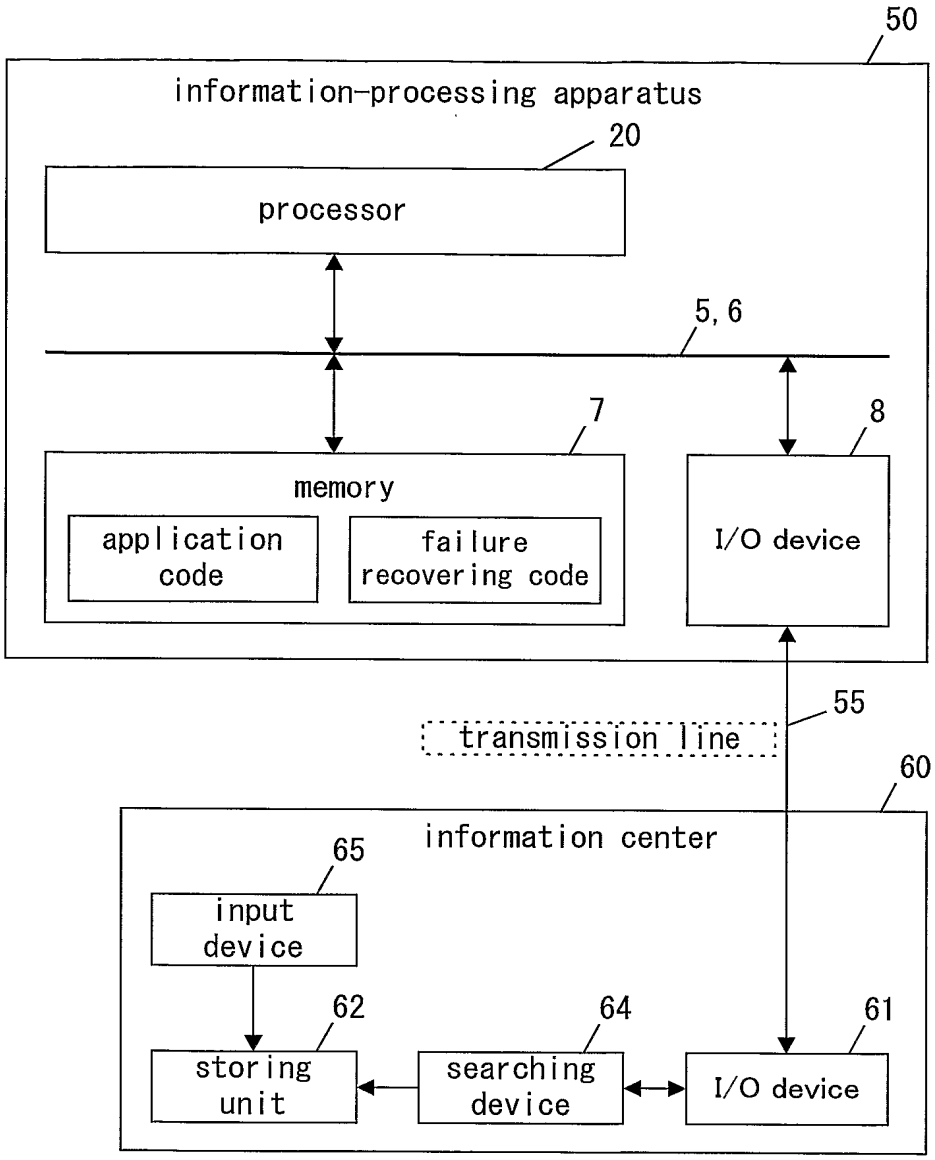
50
information-processing apparatus
20
processor
5, 6
7
8
memory
application code
failure recovering code
I/O device
55
transmission line
60
information center
65
input device
62
64
61
storing unit
searching device
I/O device

Fig. 22(a)

```
extern void read_str(char *str);

int read_int()
{
    char buf[32];

    read_str(buf);

    return atoi(buf);
}
```

source code

Fig. 22(b)

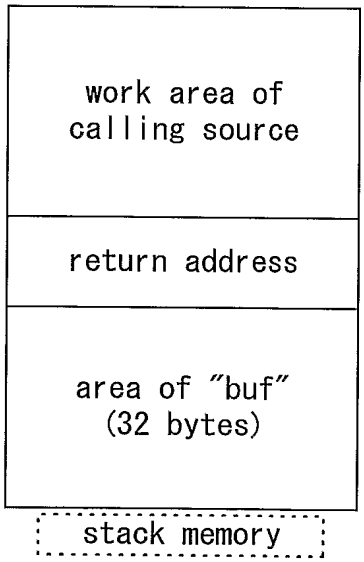
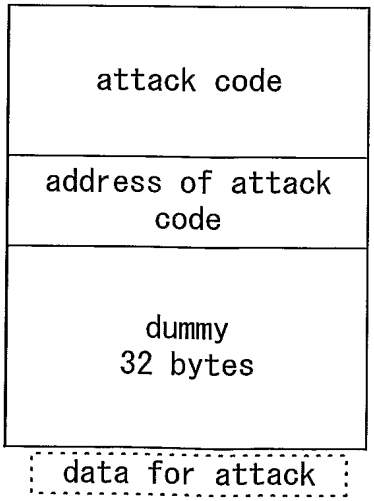
Fig. 22(c)